



EducaCiência FastCode

Fala Galera,

- Artigo: 43/2021 Data: Fevereiro/2021
- Público-alvo: Desenvolvedores – Iniciantes
- Tecnologia: Java
- Tema: Artigo 43 - SpringBoot JPA MySql Get
- Link: <https://github.com/perucello/DevFP>

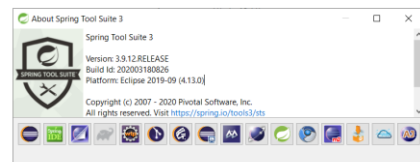
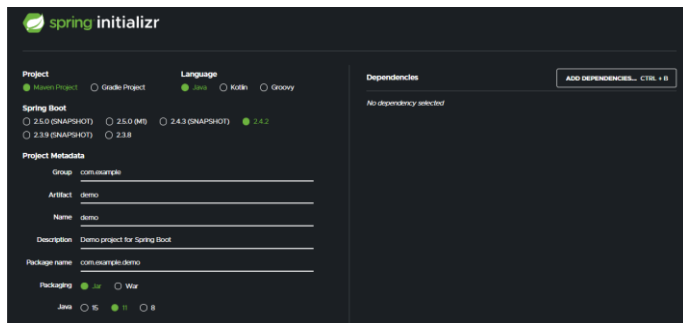
Neste artigo, abordaremos Spring Boot e iremos mapear o Método Get com repositório JPA Repository e Banco de Dados MySql.

Traremos uma série de artigos onde exploraremos os métodos de maneira individual até chegarmos em uma API com todos os métodos.

Lembrando que os fins são didáticos !

Para este ambiente , utilizaremos do Banco de Dados MySql , deixaremos um script de criação do Banco anexado ao nosso artigo.

Criaremos nosso projeto utilizando do link - <https://start.spring.io/> e abriremos na IDE Spring Tool Suíte 3 “STS”.





Com nosso Projeto já aberto no STS , iremos preparar nosso arquivo “pom”, ou seja, nossas dependências que iremos trabalhar no projeto.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.2</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.project.jpa.mysql</groupId>
  <artifactId>SpringBoot-JPA-Mysql</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>SpringBoot-JPA-Mysql</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>1.8</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <scope>runtime</scope>
    </dependency>

    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
    </dependency>

    <dependency>
      <groupId>javax.validation</groupId>
      <artifactId>validation-api</artifactId>
      <version>1.1.0.Final</version>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
```

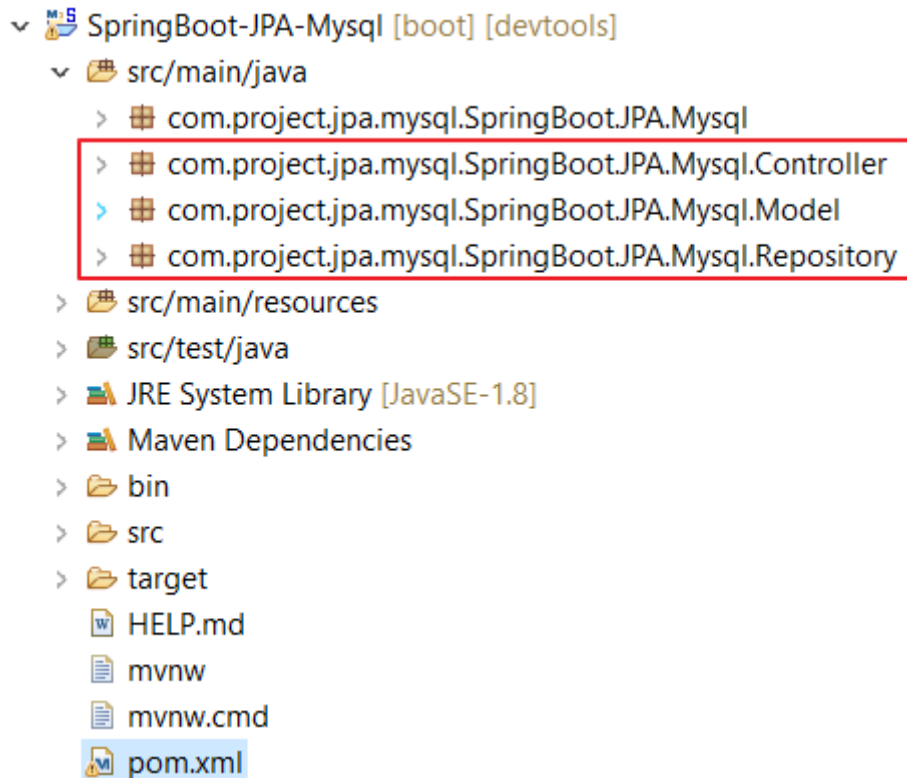


```
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>
```

Feito isso, vamos atualizar nosso Maven – para isso, basta executar “Maven Install”.

Vamos preparar agora nossos “pacotes” como demonstrados abaixo:



Feito isso, vamos criar nossas classes sendo:



Vamos detalhar:

- a) **Controller** – neste pacote teremos nossa classe “ClientesController” onde será escrito nosso código que manipularemos os métodos CRUD e nosso endpoint;
- b) **Model** – neste pacote teremos nossa classe “Cliente” onde criaremos a estrutura da nossa tabela Cliente no Banco de Dados
- c) **Repository** – teremos nossa “interface” que se estenderá da classe “Cliente” e receberá nosso Repository JPA.

Primeiramente, vamos criar a estrutura da nossa tabela, como a seguir:

```
package com.project.jpa.mysql.SpringBoot.JPA.Mysql.Model;
```

```
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.validation.constraints.NotNull;
```

```
@Entity  
public class Cliente {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
  
    private String nome;  
  
    @NotNull  
    private String email;  
  
    public Cliente() {  
        super();  
    }  
  
    public Cliente(Long id, String nome, String email) {  
        super();  
        this.id = id;  
        this.nome = nome;  
    }  
}
```



```
        this.email = email;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((id == null) ? 0 : id.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Cliente other = (Cliente) obj;
        if (id == null) {
            if (other.id != null)
                return false;
        } else if (!id.equals(other.id))
            return false;
        return true;
    }
}
```

Agora, podemos estender nossa interface, para isso , abra o pacote Repositório e clique em Clientes.java e insira o seguinte código:

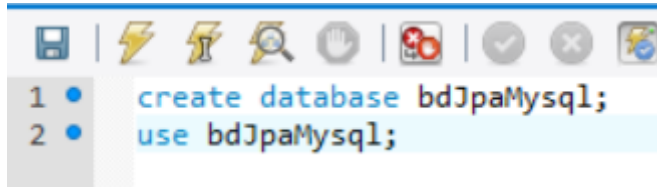
```
package com.project.jpa.mysql.SpringBoot.JPA.MySql.Repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.project.jpa.mysql.SpringBoot.JPA.MySql.Model.Cliente;

public interface Clientes extends JpaRepository<Cliente, Long> {
}
```

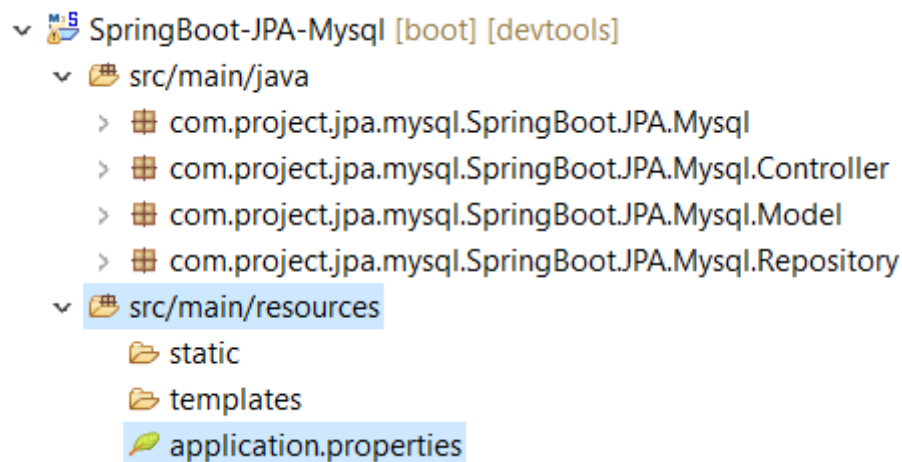


Feito isso, temos a estrutura do nosso banco de dados (tabela) pronta e agora , agora precisamos criar nosso Banco de Dados no Mysql, para isso, abra seu Workbench e insira os seguintes códigos e execute.



```
1 • create database bdJpaMysql;
2 • use bdJpaMysql;
```

Com o Banco de Dados criado, vamos preparar a conexão do nosso Projeto, sendo assim, acesse application.properties pela rota src/main/resources e insira os seguintes códigos:



```
SpringBoot-JPA-Mysql [boot] [devtools]
├── src/main/java
│   ├── com.project.jpa.mysql.SpringBoot.JPA.Mysql
│   ├── com.project.jpa.mysql.SpringBoot.JPA.Mysql.Controller
│   ├── com.project.jpa.mysql.SpringBoot.JPA.Mysql.Model
│   └── com.project.jpa.mysql.SpringBoot.JPA.Mysql.Repository
└── src/main/resources
    ├── static
    ├── templates
    └── application.properties
```

```
spring.datasource.url=jdbc:mysql://localhost:3306/bdJpaMysql?useTimezone=true&serverTimezone=UTC
&useSSL=false
spring.datasource.username=root
spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
```

Com a nossa estrutura do Banco de Dados criado, (entidade e repositório) agora, podemos focar em nosso CRUD.

Para este trabalho, abra o Script **ClientesController** que está no pacote Controller e insira os seguintes códigos:

Com isso , temos nosso Post do CRUD pronto onde os métodos que manipularemos será o seguinte:

```
package com.project.jpa.mysql.SpringBoot.JPA.Mysql.Controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.project.jpa.mysql.SpringBoot.JPA.Mysql.Model.Cliente;
```

```
import com.project.jpa.mysql.SpringBoot.JPA.MySql.Repository.Clientes;

@RestController
@RequestMapping("/api/JPA/MySql/clientes")
public class ClientesController {

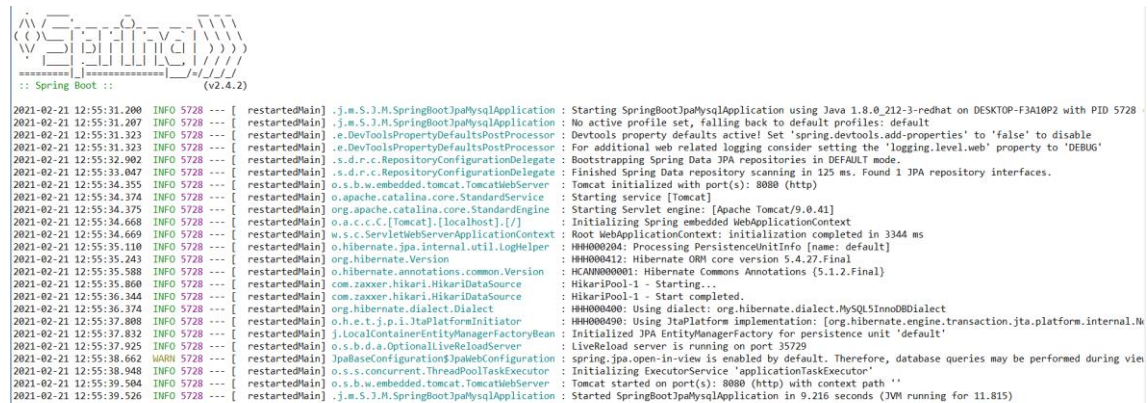
    @Autowired
    private Clientes clientes;

    @GetMapping
    public List<Cliente> listar(){
        System.out.println("Quantidade de Registros de Clientes : " + clientes.count());
        return (clientes.findAll());
    }
}
```

⇒ **Get** – este método irá realizar um Select dos dados em nossa tabela.

Agora, basta iniciarmos nossa “Aplicação”

⇒ Run As \ Spring Boot App



```

:: Spring Boot ::
:: (v2.4.2)

2021-02-21 12:55:31.200 INFO 5728 --- [ restartedMain] .j.m.s.j.m.SpringBootJpaMySqlApplication : Starting SpringBootJpaMySqlApplication using Java 1.8.0_212-3-redhat on DESKTOP-F3A10P2 with PID 5728
2021-02-21 12:55:31.207 INFO 5728 --- [ restartedMain] .j.m.s.j.m.SpringBootJpaMySqlApplication : No active profile set, falling back to default profiles: default
2021-02-21 12:55:31.323 INFO 5728 --- [ restartedMain] o.e.DevToolsPropertyDefaultsPostProcessor : DevTools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable
2021-02-21 12:55:31.323 INFO 5728 --- [ restartedMain] o.e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' property to 'DEBUG'
2021-02-21 12:55:32.902 INFO 5728 --- [ restartedMain] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2021-02-21 12:55:33.047 INFO 5728 --- [ restartedMain] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 125 ms. Found 1 JPA repository interfaces.
2021-02-21 12:55:34.355 INFO 5728 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-02-21 12:55:34.374 INFO 5728 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-02-21 12:55:34.375 INFO 5728 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.41]
2021-02-21 12:55:34.668 INFO 5728 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2021-02-21 12:55:34.669 INFO 5728 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: Initialization completed in 3344 ms
2021-02-21 12:55:35.110 INFO 5728 --- [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HH000204: Processing PersistenceUnitInfo [name: default]
2021-02-21 12:55:35.243 INFO 5728 --- [ restartedMain] org.hibernate.Version : HH0000412: Hibernate ORM core version 5.4.27.Final
2021-02-21 12:55:35.588 INFO 5728 --- [ restartedMain] org.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations (5.1.2.Final)
2021-02-21 12:55:35.860 INFO 5728 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2021-02-21 12:55:36.344 INFO 5728 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2021-02-21 12:55:36.374 INFO 5728 --- [ restartedMain] org.hibernate.dialect.Dialect : HH0000490: Using dialect: org.hibernate.dialect.MySQL5InnoDBDialect
2021-02-21 12:55:37.808 INFO 5728 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HH0000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.W
2021-02-21 12:55:37.832 INFO 5728 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2021-02-21 12:55:37.925 INFO 5728 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2021-02-21 12:55:38.662 INFO 5728 --- [ restartedMain] jpaBaseConfigurationJpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view
2021-02-21 12:55:38.948 INFO 5728 --- [ restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2021-02-21 12:55:39.504 INFO 5728 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-02-21 12:55:39.526 INFO 5728 --- [ restartedMain] .j.m.s.j.m.SpringBootJpaMySqlApplication : Started SpringBootJpaMySqlApplication in 9.216 seconds (JVM running for 11.815)

```

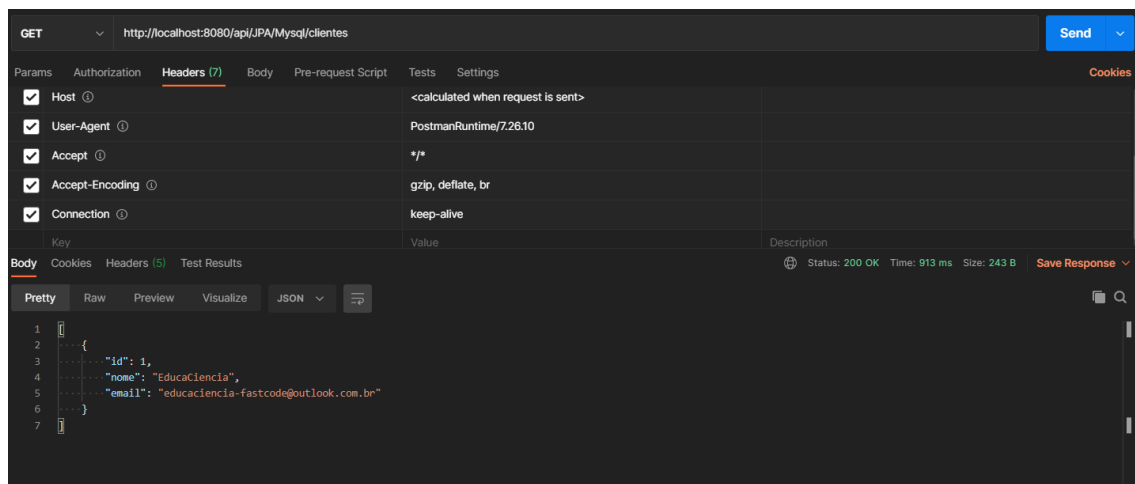
Nossa aplicação se iniciou como esperado.

Agora podemos testar , manipulando via Postman, para isso abra seu aplicativo e vamos testar o método que criamos em nossa API.

Vamos inserir o Endpoint de acesso e manipulá-lo como abaixo:

⇒ Lembrando que no artigo 42/2021 nós realizamos já o Post , sendo assim, já temos um dado inserido, portanto iremos manipular o endpoint para o método Get.

⇒ **Retorno esperado quando se obtém sucesso é 200 neste caso.**



Nossa API funcionou como nossa proposta, sendo assim finalizamos este artigo, onde os códigos estão disponíveis no GitHub para consumo.

Até mais !
Espero ter ajudado !

